



# Semantic MediaWiki

## An Introduction



Markus Krötzsch  
University of Oxford

SMWCon Fall 2012



October 24, 2012

# About these slides ...

These slides have been presented at  
[SMWCon Fall 2012](#) as part of a 90min tutorial.

The slides are published under Creative Commons  
CC-By 3.0, with the exception of images on page 41,  
45, 57, 62 and 69, which retain the copyright of their  
respective owners.

# Semantic MediaWiki (SMW)

... is a **content management software** for

- Richly formatted hypertext documents
- Structured data

... based on the “**Wiki Way**”:

- User-governed
- Collaborative
- Easy



# Semantic MediaWiki (SMW)

... is **based on MediaWiki** (MW):

- Wiki engine for managing hypertext
- Very popular
  - Wikipedia, Wikia, >100,000 sites
- User interface in hundreds of languages
- Free, Open Source, open development
  - Some full-time developers (Wikimedia staff)
- Written in PHP and JavaScript, main DB MySQL

# Semantic MediaWiki (SMW)

... is implemented as a **MediaWiki extension**:

- Modular add-on
  - Fully compatible with current and recent MW
- Free, Open Source, open development
  - Driven by volunteers: no staff
  - Integrated with MW development
  - Current maintainers: MK, Jeroen De Dauw
- Written in PHP and JavaScript, main DB MySQL
  - RDF DBs supported as secondary storage

# Semantic MediaWiki (SMW)

... is **well-established and stable**

- Created in 2005
- Used on hundreds of sites (impossible to count)
- Extensive documentation in many languages
  - <http://semantic-mediawiki.org/>
  - Current maintainer: Karsten Hoffmeyer
- Twice-yearly user conference SMWCon
- Organisational support by OSDA (charity)
- One of the biggest MW extensions
  - Code, people, commit activity, user community

# MediaWiki



WIKIPEDIA  
The Free Encyclopedia

[Main page](#)  
[Contents](#)  
[Featured content](#)  
[Current events](#)  
[Random article](#)  
[Donate to Wikipedia](#)

▼ [Interaction](#)  
[Help](#)  
[About Wikipedia](#)  
[Community portal](#)  
[Recent changes](#)  
[Contact Wikipedia](#)

► [Toolbox](#)  
[Print/export](#)

▼ [Languages](#)  
★ [Afrikaans](#)  
[Alemannisch](#)  
[አማርኛ](#)  
[العربية](#)  
[Aragonés](#)  
[Azərbaycanca](#)  
[Bân-lâm-gú](#)  
[Беларуская](#)  
[Беларуская \(тарашкевіца\)](#)  
[Български](#)  
[Boarisch](#)  
[Bosanski](#)  
[Brezhoneg](#)  
[Català](#)

[Create account](#) [Log in](#)

Article [Talk](#)

[Read](#) [Edit](#) [View history](#)

## Cologne

From Wikipedia, the free encyclopedia

Coordinates: 50°57′N 6°58′E﻿ / ﻿50.95°N 6.97°E﻿ / 50.95; 6.97

*"Köln" redirects here. For other uses, see [Köln \(disambiguation\)](#).*

*This article is about the German city. For the perfume, see [Eau de Cologne](#). For other uses, see [Cologne \(disambiguation\)](#).*

**Cologne** (English pronunciation: /kəˈloʊn/, German: *Köln* [kœln], Kölsch: *Kölle* [ˈkœlə]) is Germany's fourth-largest city (after Berlin, Hamburg and Munich), and is the largest city both in the German Federal State of North Rhine-Westphalia and within the Rhine-Ruhr Metropolitan Area, one of the major European metropolitan areas with more than ten million inhabitants.

Cologne is located on both sides of the Rhine River. The city's famous [Cologne Cathedral](#) (*Kölner Dom*) is the seat of the [Catholic Archbishop of Cologne](#). The [University of Cologne](#) (*Universität zu Köln*) is one of Europe's oldest and largest universities.<sup>[2]</sup>

Cologne is a major cultural centre of the [Rhineland](#) and has a vibrant arts scene. Cologne is home to more than 30 museums and hundreds of galleries. Exhibitions range from local ancient Roman archeological sites to contemporary graphics and sculpture. The [Cologne Trade Fair](#) hosts a number of trade shows such as [Art Cologne](#), [imm Cologne](#), [Gamescom](#) and the [Photokina](#).

### Contents [hide]

- History
  - 1.1 Roman Cologne
  - 1.2 Middle Ages
  - 1.3 From 19th century until World War II
  - 1.4 World War II
  - 1.5 Post-war Cologne until today
  - 1.6 Post-reunification
- Geography
  - 2.1 Districts
  - 2.2 Climate
  - 2.3 Flood protection
- Demographics
- Government
  - 4.1 Political traditions and developments
  - 4.2 Mayor
  - 4.3 Elections
  - 4.4 Status of other districts

*Köln*  
Cologne



View of Cologne, July 2006



**Cologne within North Rhine-Westphalia** [show]

# MediaWiki: A Tool for Managing Wikitext

- Wiki pages are formatted hypertext documents
- “Wikitext” input converted to HTML output:

```
[[Link to wiki-page]]  
[[Some page|link with custom text]]  
[http://example.org Link to an outside URL]  
==Header 1==  
===Header 2=== (etc.)  
'italics'  
'''bold'''  
* bulleted list  
# numbered list  
: indentation
```

# MediaWiki: Advanced formatting features

- Rich set of features for controlling content display
- Many HTML-like features supported (<sub>, style, ...)
- Advanced formatting elements:
  - Images
  - Tables

→ See online documentation for details

# MediaWiki: Page names

- Page names consists of multiple parts:

**Namespace:**Title/Subpagetitle

- Examples:
  - User:Denny/Tests
  - Köln
  - Talk:Köln
  - 2001: A Space Odyssey
  - Std::out
  - /dev/null

# MediaWiki: Page names

- Page names consists of multiple parts:

**Namespace:**Title/**Subpagetitle**

- Examples:
  - User:Denny/Tests
  - Köln
  - Talk:Köln
  - 2001: A Space Odyssey
  - Std::out
  - /dev/null

# MediaWiki: Namespaces

- Prefixes, separated from title by colon “:”
- **Not all prefixes that end in “:” are namespaces!**
  - Available prefixes provided by MW, more can be added in configuration
  - Default: *Main* (empty), *User*, *Category*, *Template*, *Help*, *MediaWiki*, *File*, *Special*, *Project* (sitename)
  - Purpose is to distinguish basic “content types”
- Namespaces can have aliases (e.g. *File:* and *Image:*)
- *Special:* is for pages that with fixed functionality, no editing
- Every namespace has a talk version: *Talk*, *User talk*, ...

# MediaWiki: Subpages

- Postfixes, separated from title by slash "/"
- **Not all postfixes after "/" are subpages!**
  - Enabled for certain namespaces
    - By default only for User and all Talk pages
- Often not appropriate for organising pages (rigid, hierarchical content structure)
- Small difference to pages with "/" in title
  - For example when moving pages

# MediaWiki: Categories

- Preferred way to organise pages in MediaWiki
- “Links” to Category pages mean: “page is in category”
  - Example: [[Category:City]] on page of Cologne
- Categories can be described by pages
- Category “Links” on Category pages mean: “page is subcategory of”
  - Example: [[Category:Settlement]] on Category:City
- Category “hierarchy” can be any graph (multi-inheritance, cycles, ...)
- Used for browsing, but not for search

# MediaWiki: Redirects

- A page can redirect to any other
- It should contain one line only:

`#REDIRECT [[Target page name]]`

- Very useful for creating synonyms and search hints

# MediaWiki: Templates

- Transclusion:  
Insert the content of one page into another
- Parameters can be given:  
Use given values to substitute place holders in transcluded page
- Wikitext to transclude a page with parameters:  
`{{Page to transclude|parameter1=value1| ... }}`  
(Default: assume "Template:" as namespace for page)
- Wikitext for place holders: `{{{parameter1}}}`

# Parser functions, tags, switches, variables

- Special syntax elements with hard-coded functions
- Parser functions: Usually `{{#function:param1|...}}`
  - Examples: `#if`, `#ifeq`, `#switch`, `#replace`
- Parser tags: Like HTML or XML tags in `< >`
  - Examples: `<nowiki>`, `<noinclude>`, `<includeonly>`
- Behaviour switches: control some page effects
  - Examples: `__NOTOC__`, `__SHOWFACTBOX__` (SMW)
- Variables: insert some data into wikitext
  - Examples: `{{PAGENAME}}`, `{{CURRENTDAY}}`

# Semantic MediaWiki



# Semantic MediaWiki: Main Features

- Store & manage data
- Browse & view data
- Search pages based on their stored data
- Format and display data
- Export data

## What is *data*?

# Semantic MediaWiki: Data model

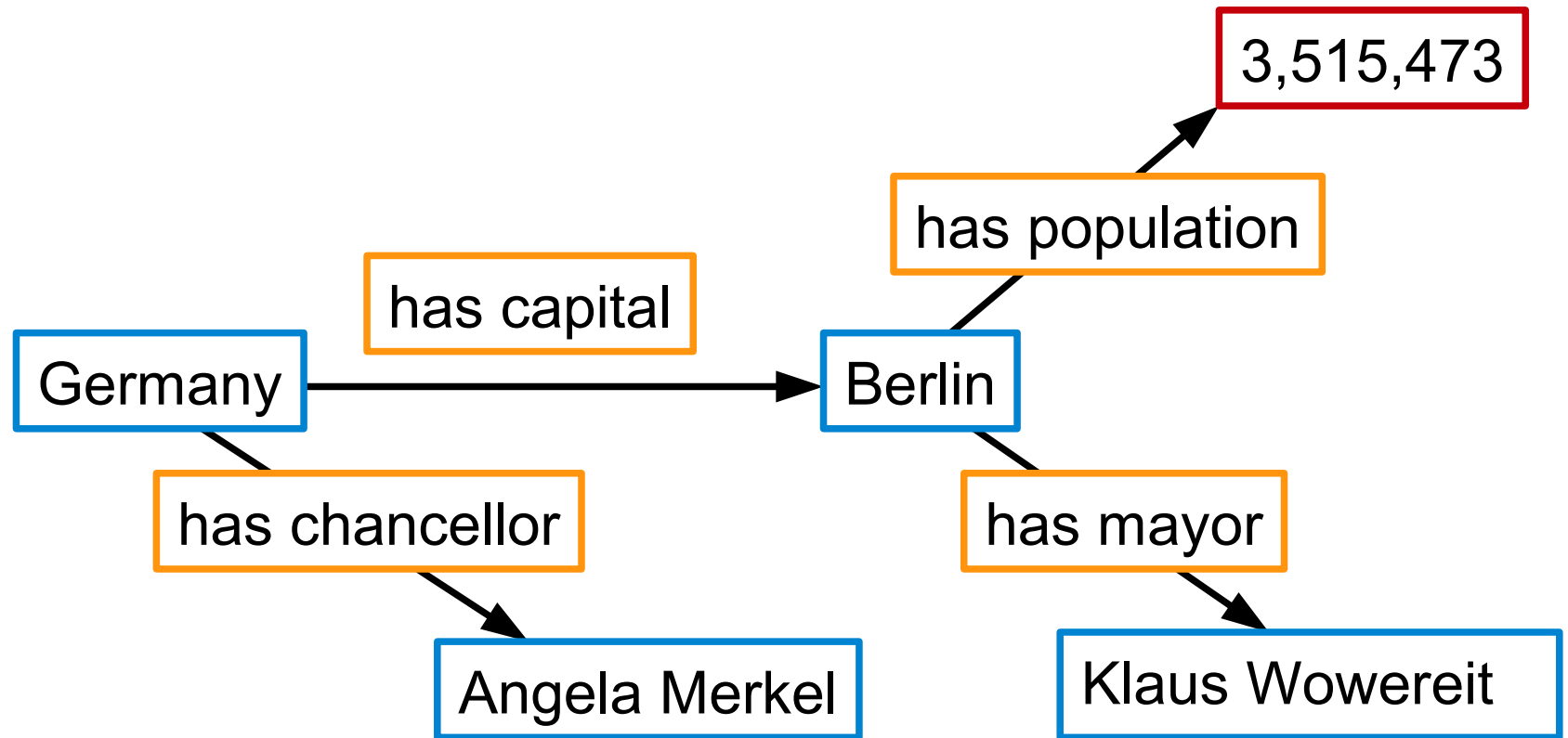
- Data: **property-value pairs assigned to pages**
- Examples (page – property – value):

Cologne	has population	1.017.155
Germany	has capital	Berlin
Spinoza	born on	24 Nov 1632

- Pages can be related to pages (see *has capital*)

# Semantic MediaWiki: Data model

- Think of SMW data in a graphical way:



# Semantic MediaWiki: Entering data

- Data is edited on the page that it is about, e.g. Germany:



- Two basic methods:

`[[has capital::Berlin]]`      Displays like as link `[[Berlin]]`

`{{#set: has capital=Berlin}}`      Not displayed

# Semantic MediaWiki: Properties

- “Germany” is a wikipage, “Berlin” is a wikipage ...  
... what is “has capital”?



→ A wikipage in the namespace *Property*:

Property:has capital

# Semantic MediaWiki: Properties

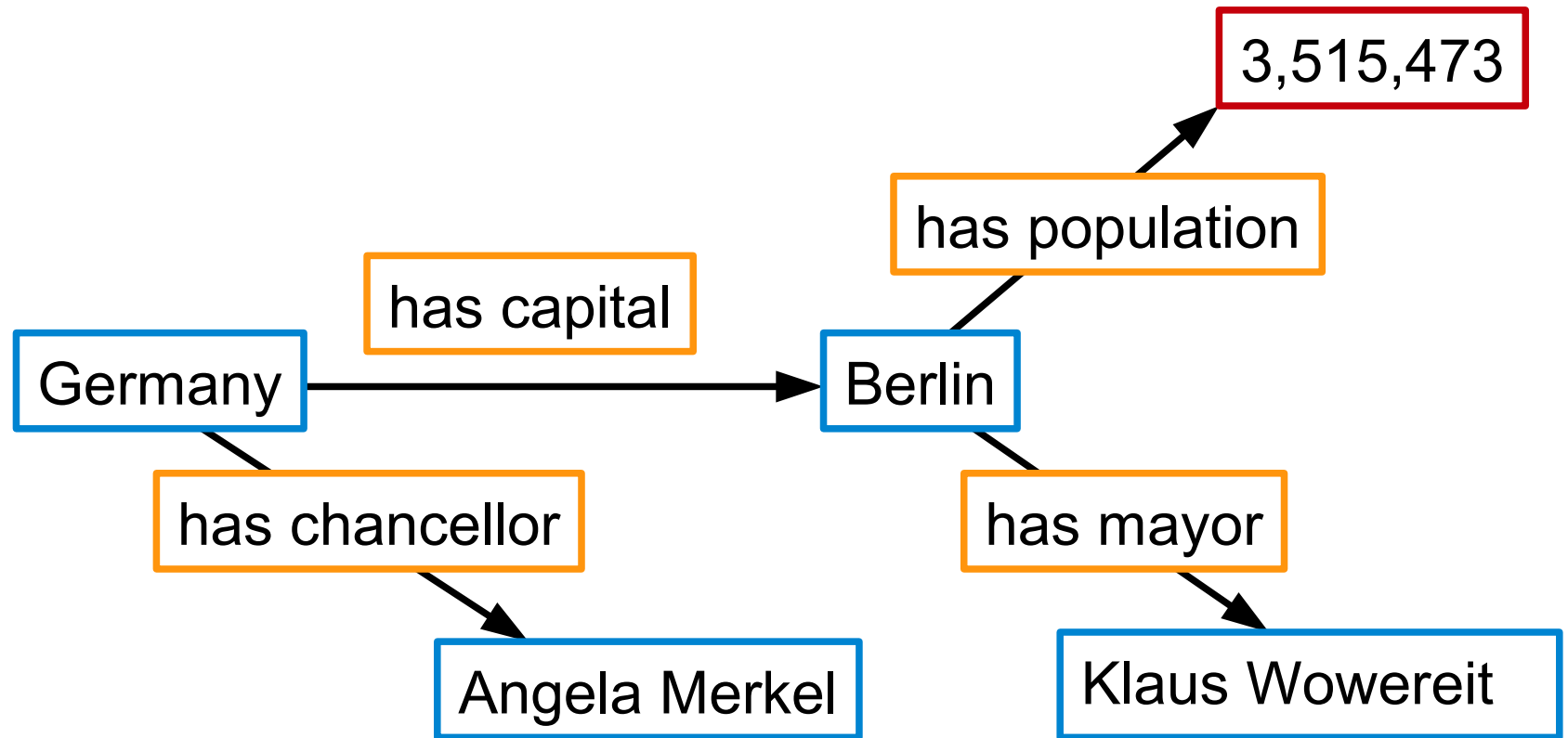
- Property pages have two main uses
  - Document properties (how/when to use them?)
  - Declare properties (how should SMW treat them?)
- Main declaration: the **datatype** of the property
  - Specified using property “has type”
  - Common types: Number, Date, Wikipage (default), Text, ...
  - Default type (if undeclared): Wikipage

# Semantic MediaWiki: Property Datatypes

- Datatypes control input and output (and more ...)
- Output:
  - [[has capital::Germany]] appears as link
  - [[has population::1000]] appears as number
- Input:
  - [[my property::1000]] vs. [[my property::1,000]]
    - Different value if “*my property*” has type Wikipage
    - Same value if “*my property*” has type Number

# Semantic MediaWiki: Data model

- It is now clear how to enter this data



# Semantic MediaWiki: Retrieving data

- How can we view the data we enter?
  - Special:Browse – view all data for a page
  - Property:Name – view all data for property
  - Special:Types – view all properties of some type
  - Special:Properties – view all properties
  - And various others (not quite as important)
- Factbox shows all data about one page
  - Below edit field in editing preview
  - At page bottom on some or all pages

# Semantic MediaWiki: Retrieving data

- How can we embed data into wikipages?  
→ Parser function `#show` can display property values
- Examples:

`{{#show:Berlin|?has population}}` – show this number  
`{{#show:Germany|?has capital}}` – show this page (link)

→ Re-use data without re-entering it

# Semantic MediaWiki: Querying data

- How can we find pages based on their data  
→ Parser function #ask can answer many queries
- Examples:

`{{#ask: [[Category:City]] }}` – show pages with category

`{{#ask: [[has mayor::Klaus Wowereit]] }}`

– show pages with this data

`{{#ask: [[Category:City]] [[has mayor::Klaus Wowereit]] }}`

– show pages with both

# The #ask Query Language (AQL)

- Main idea: to ask for a page with some annotation, use this annotation (as in examples just seen)
  - `[[Category:City]]`
  - `[[has mayor::Klaus Wowereit]]`
- Some query conditions cannot work quite like this:
  - `[[has population::+]]` – pages with *some* population
  - `[[User:+]]` – pages in namespace User
  - `[:Category:+]` – pages in namespace Category

# AQL Disjunction and Conjunction

- Keyword “OR” can be used to allow alternatives:  
`{{#ask: [[Category:City]] OR [[Category:Town]] }}`
- Default: find pages that satisfy all conditions given (this would be “AND”)

# AQL Inverse Properties

- One can invert the direction of Wikipage properties

```
{{#ask: [[has capital-::Germany]] }}
```

Here “has capital-” has the meaning “is capital of”.

- This can be done in all SMW interfaces that take properties, but not when adding data to a page

# AQL Property Values

- We have already seen two types of property values
  - `[[has mayor::Klaus Wowereit]]` – concrete value
  - `[[has population::+]]` – any value (at least one)
- Other conditions based on comparators:
  - `[[has population::>1000]]` – greater or equal 1000
  - `[[has population::<1000]]` – less or equal 1000
  - `[[has population::!1000]]` – inequal 1000

Note: it always means “**has some value** for the property that is greater/less/not equal to 1000”

# AQL Subqueries

- A more elaborate form of property values for Wikipages: the pages that match another query
- Example: “Countries with small capital cities”

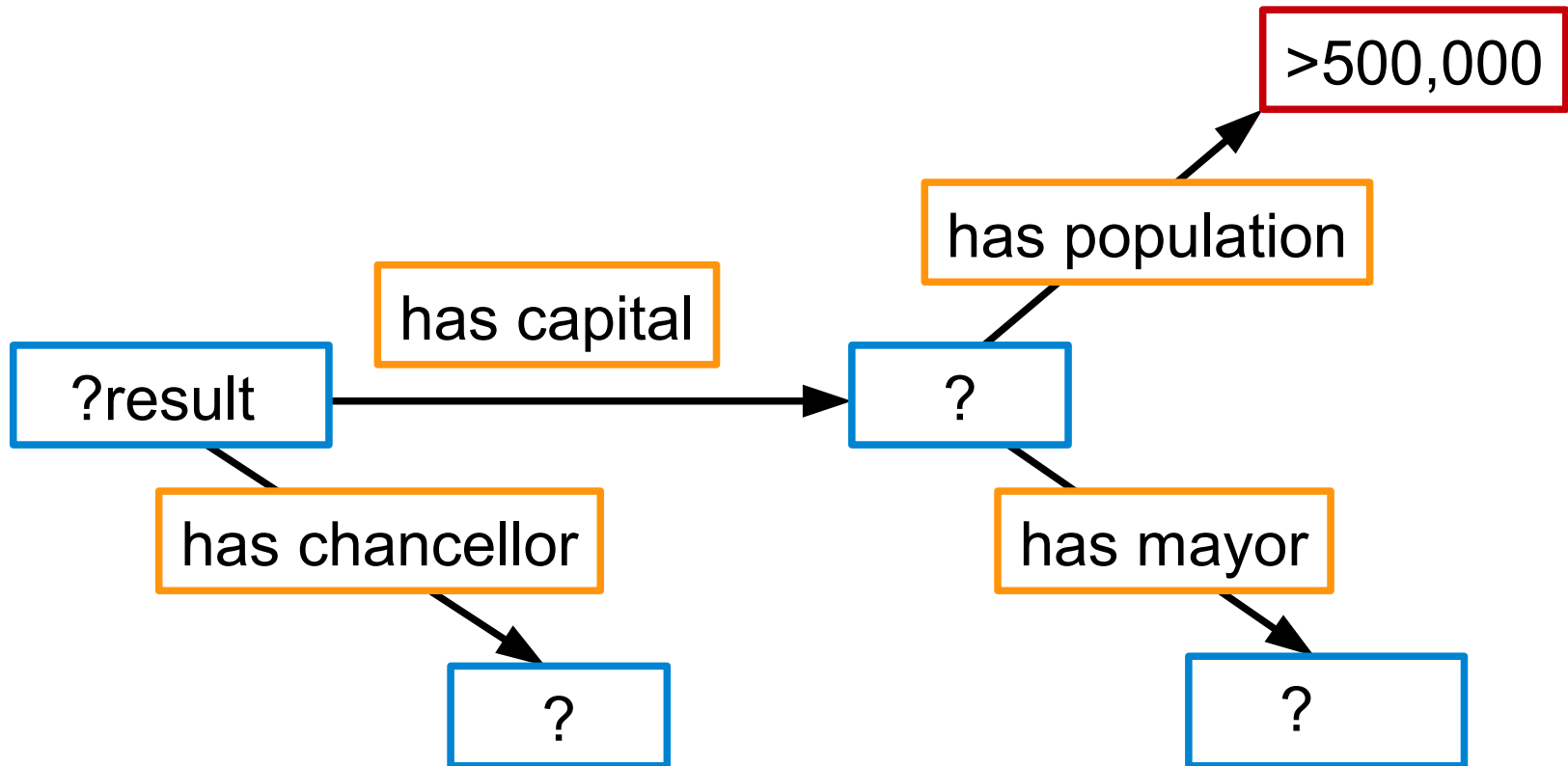
```
{{#ask: [[has capital::  
      <q>[[has population::<500,000]]</q> ]]]}}
```

- Property chains like this can be abbreviated:

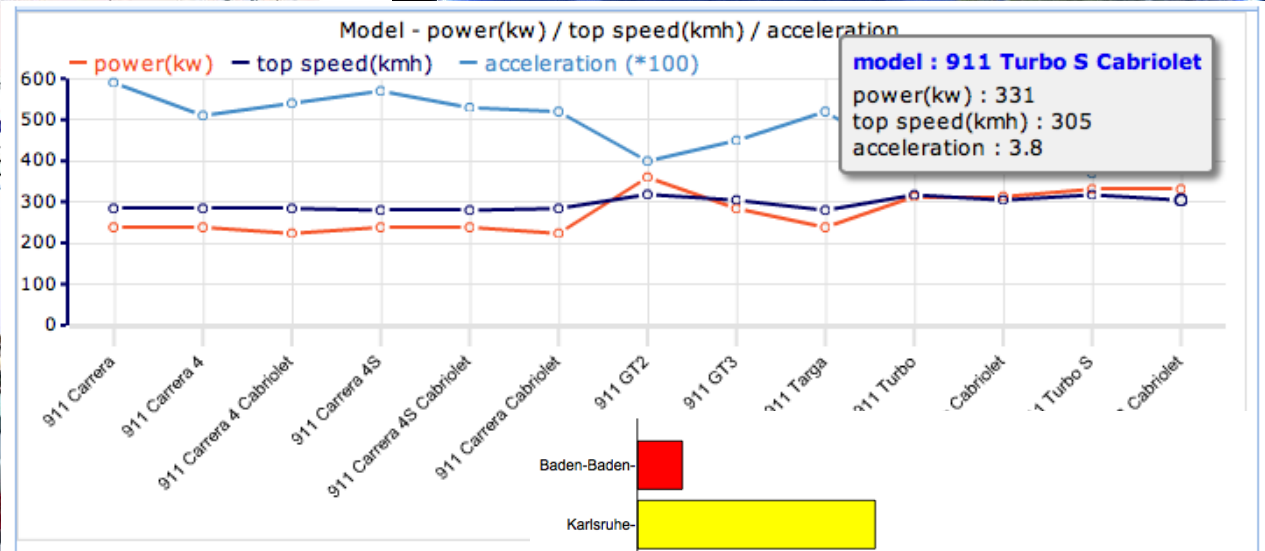
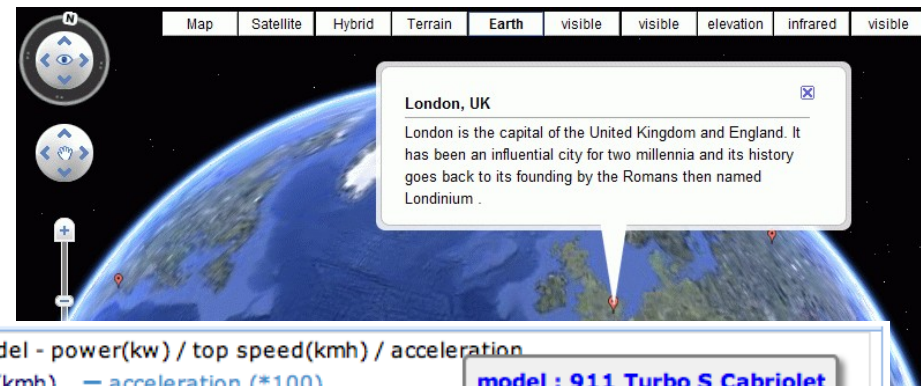
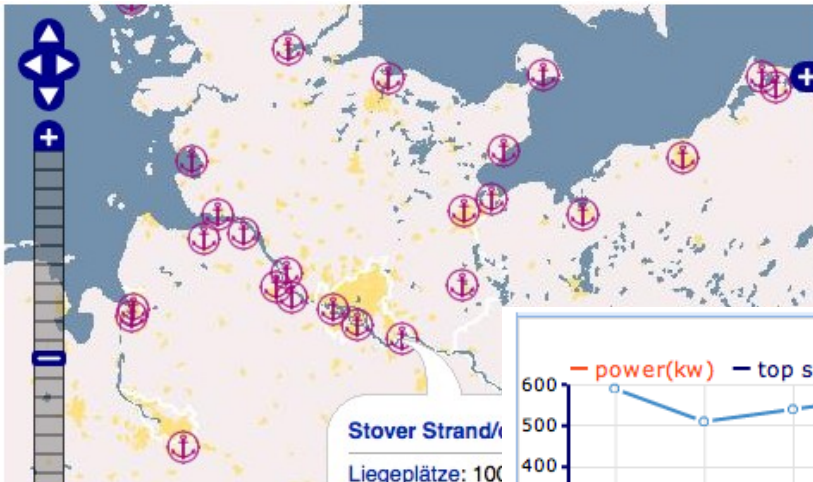
```
{{#ask: [[has capital.has population::<500,000]] }}
```

# AQL Queries as graphs

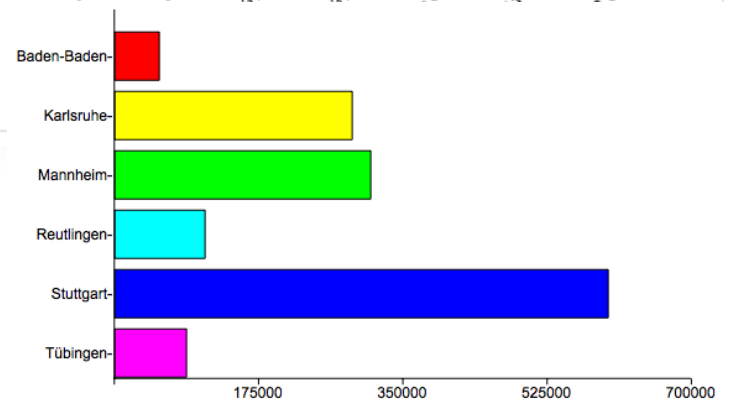
- Think of query conditions as graphs:



# Displaying Query Results



Mercedes-Benz E-Class coupe (US) Rear view of the E-Class coupe



■ Mercedes Benz W212 : Mercedes-Benz E-Class coupe (US) Rear view of the E-Class coupe

■ Thumbnail : Mercedes-Benz E 350 CGI Coupé Heck.JPG

# Additional Data in Query Results

- Printout requests, given as parameters  
“?property name” in #ask
- Example:

```
{{#ask: [[Category:City]]  
  |?has population  
  |?has mayor  
}}
```

- Displays as a table with three columns

# Setting a query format

- Format defined by parameter “format”, e.g.

```
{{#ask: [[Category:City]]  
  |?has population  
  |?has mayor  
  |format=ol  
}}
```

→ Displays results as an ordered list

# Many query formats

- List results on page: list, ol, ul, table, ...
- Export query results: rss, json, csv, dsv, ...
- Format results with template: template
- Embed result pages instead: embed
- Maps
- Fancy interactive formats: timelines, data browsers, slide shows, ...

Partly in extension Semantic Result Formats (SRF) and Semantic Maps (SM)

→ it is not hard to add your own ...

# Extensions



# Essential SMW extensions

- **Semantic Forms**

Input forms for strongly templated pages

- **Maps and Semantic Maps**

Manage and query data about geographic locations

- **Semantic Result Formats**

Many output formats

# Popular SMW extensions

- **Semantic Drilldown**

A faceted browser – performance can be a problem

- **Semantic Watchlist**

Get notified when data changes

- **Semantic Internal Objects**

Create auxiliary objects – we will look into this later

# Semantic Bundle

- Package with SMW and various helpful extensions (including some that are not related to SMW)
- Other extensions one should install include all spam fighting extensions (SMWCon Fall 2012 has another tutorial about this)

# Advanced Data Encoding in SMW

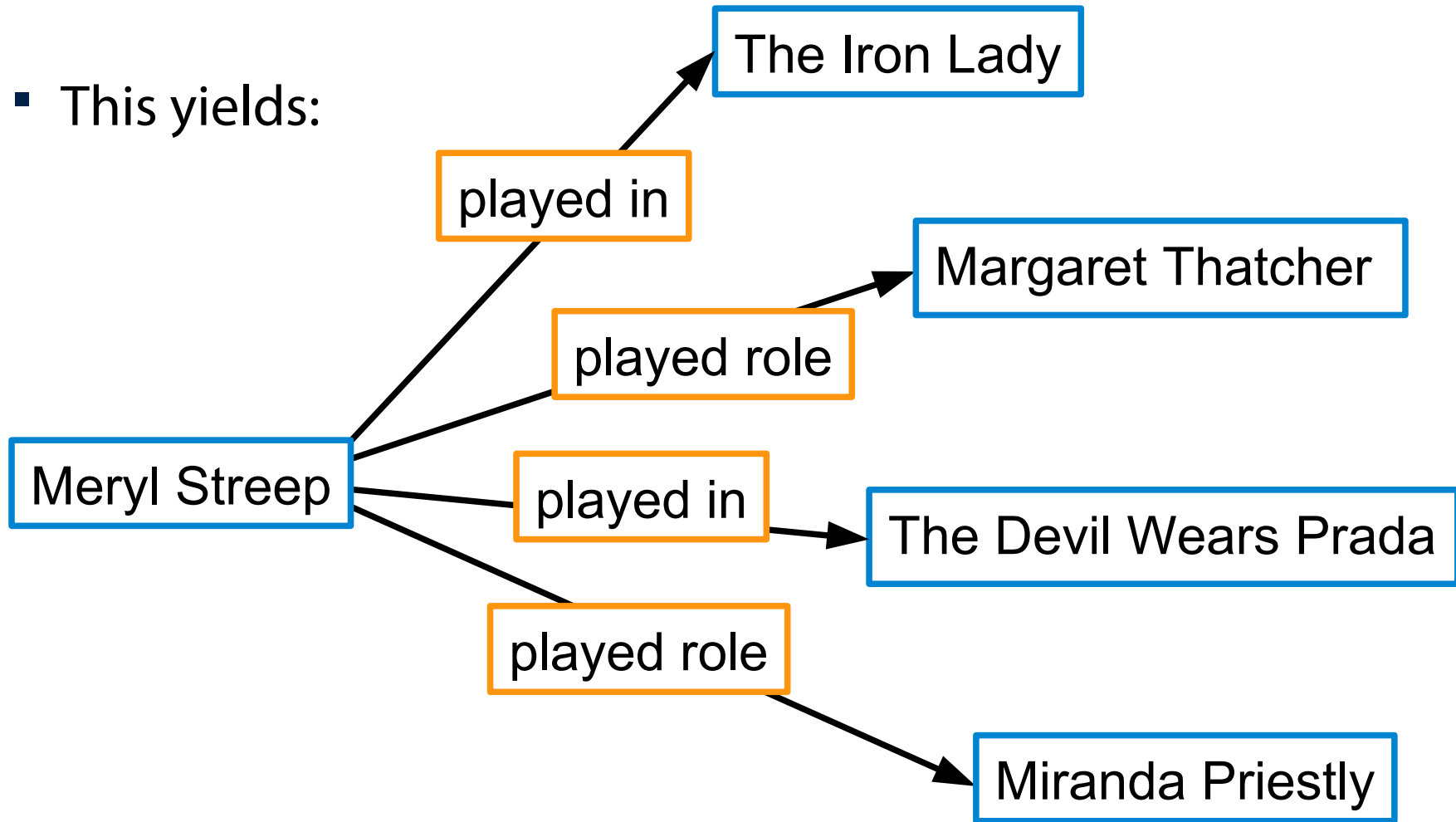


# Not all data is a property value

- Example:  
“Meryl Streep played *Margaret Thatcher* in *The Iron Lady*”
- Properties “played role” and “played in”?  
Meryl Streep played in *The Iron Lady*  
Meryl Streep played role *Margaret Thatcher*
- But also:  
Meryl Streep played in *The Devil Wears Prada*  
Meryl Streep played role *Miranda Priestly*

# Not all data is a property value

- This yields:



# Not all data is a property value

- Example:  
“Meryl Streep played *Margaret Thatcher* in *The Iron Lady*”
- Properties “played role” and “played in”?  
Meryl Streep played in *The Iron Lady*  
Meryl Streep played role *Margaret Thatcher*
- But also:  
Meryl Streep played in *The Devil Wears Prada*  
Meryl Streep played role *Miranda Priestly*

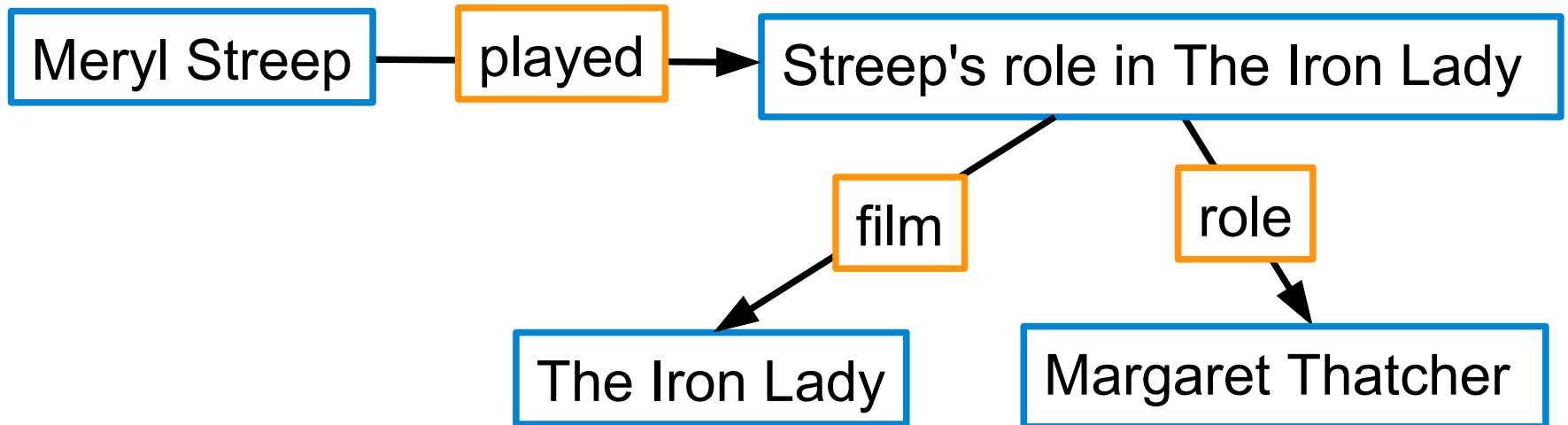
→ So did Streep play *Miranda Priestly* in *The Iron Lady*?

# Not all data is a property value

- Solution:
  - Create a new page “Streep's role in The Iron Lady” with properties:  
[[role::Margaret Thatcher]]  
[[film::The Iron Lady]]
  - Add to Meryl Streep's page:  
[[played::Streep's role in The Iron Lady]]

# Not all data is a property value

- This yields:



# Not all data is a property value

- Solution:
  - Create a new page “Streep's role in The Iron Lady” with properties:  
[[role::Margaret Thatcher]]  
[[film::The Iron Lady]]
  - Add to Meryl Streep's page:  
[[played::Streep's role in The Iron Lady]]

→ Cumbersome encoding with auxiliary pages

# Auxiliary pages as subobjects

- SMW supports the creation of “subobjects” of a page
- They can be named (optional)

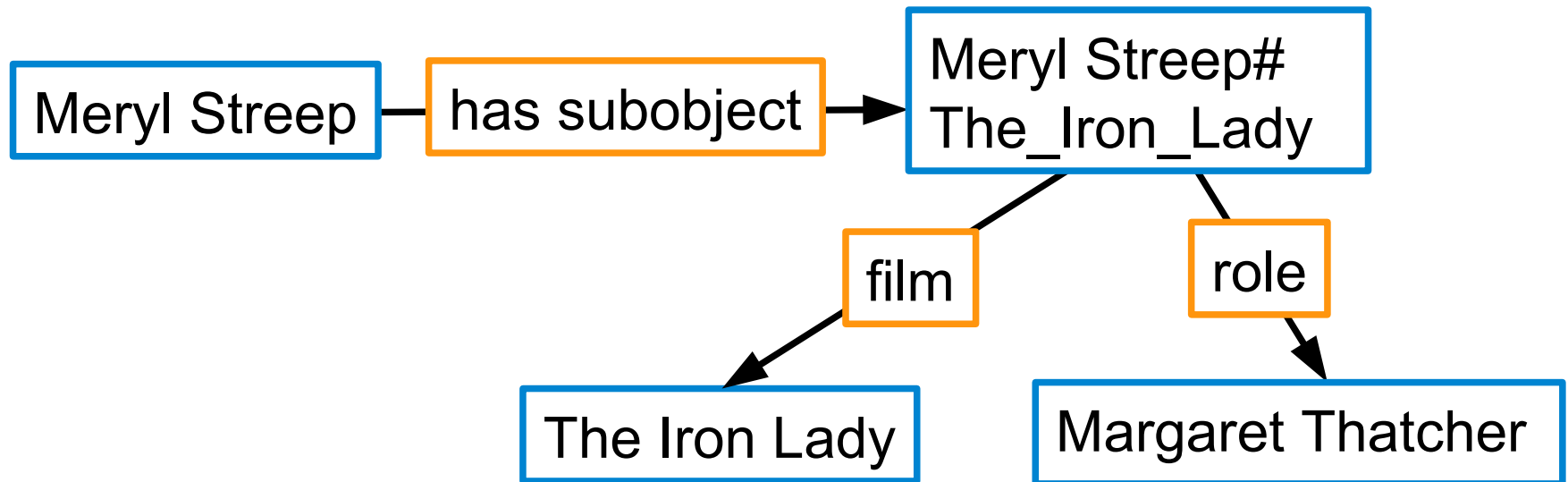
Example: Meryl Streep#The\_Iron\_Lady

- Possible with #subobject parser function, e.g.

```
{{#subobject:The Iron Lady  
|role=Margaret Thatcher  
|film=The Iron Lady  
}}
```

# Auxiliary pages as subobjects

- This yields:



# Pointing to and from subobjects

- Pointing from subobject to its page:

```
{{#subobject:The Iron Lady  
...  
|played by=Meryl Streep  
}}
```

- Pointing from (any) page to subobject:

```
[[played::Meryl Streep#The Iron Lady]]
```

# Unnamed subobjects

- One can leave away the name:

```
{{#subobject:  
  |role=Margaret Thatcher  
  |film=The Iron Lady  
}}
```

... but then one cannot point to the subobject

- This is called an *unnamed* or *anonymous* subobject

# Semantic Internal Objects extension

- Extension to create anonymous subobjects that point back to their page.

```
{{#set_internal:played by  
|role=Margaret Thatcher }}
```

is the same as

```
{{#subobject:|played by=Meryl Streep  
|role=Margaret Thatcher }}
```

# Best Practices



# Getting started with SMW

- Approach 1:  
Install SMW and rely on the *Wisdom of the Crowds*.  
→ Not known to have worked anywhere (yet?).
- Approach 2:  
Kick-start SMW usage by creating properties, input forms, data annotations, and queries that show their use.  
→ Main questions: Which data should be stored? How?  
(Fortunately, it is easy to revisit these decisions later)

# Which properties should be used?

- **Better question: What should be done with the data?**
    - Desired queries
    - Special displays (e.g. timeline)
    - Other uses for data?
- Design properties based on concrete requirements

# Collecting data with templates

- Most common style of annotation:
  - Property assignments in templates
  - Integrate display (template) and semantics (data)
  - Allows further integration with editing (Semantic Forms)
- Many parameter values can be used as property values
- Templates can be used to compute additional data
- Special case: store *absence* of data
  - SMW cannot query for pages that lack some annotation
  - Templates can add data to indicate that some information is missing (use `#if` and friends)

# Coverage and Granularity

**It is impossible to capture all human knowledge in property-value pairs.**

- How much should be covered?
- How exact and detailed should it be captured?
  - Do not attempt to build an ontology of the world
  - Start with the easy and obvious (80/20 rule)
  - Do not abuse SMW



# Keeping the balance

**Do not solve all problems with *one* template!**

- The **Template Temptation**

- Templates can use `#if` to control their behaviour based on parameters
- With SMW, templates can use `#show` and `#ask` to gather more data to control the behaviour

- Complicated, unmanageable template pages
- Slow down on page display and rendering
- Unreasonable server load

# Template-based displays

## **Avoid template-based displays, if possible.**

- Powerful tool for achieving complex formatting
- Difficult for MediaWiki to process
  - #ask can create hundreds of template calls per page
- Workaround 1: Store template rendering result as a property value that can be queried and displayed
- Workaround 2: Create PHP code to format results (not as hard as it might seem) instead of using template

# Using #ask and #show in property values

**Do not use #ask and #show in property values.**

- SMW allows things like  
`[[has mayor age::{{#show:Klaus Wowereit|?has age}} ]]`
- However, there is no update mechanism to ensure this data is updated if the query result changes
  - SMW updates page display, not stored data!
- This problem will never be solved (doing this would require SMW to process arbitrary programs, which may fail to halt.)

# Encoding Lists

## **Encoding lists is difficult.**

- Templates do not support list parameters to start with
- Semantic Forms uses two solutions:
  - the `#array` parser function to create annotations from a comma-separated list of values
  - Multiple templates are used to encode lists of more complex data structures

# Encoding Lists

## **Encoding lists is difficult.**

- Order information is hard to capture
- Main approach: make one property for every position
  - Example: author1, author2, author3, ...
    - similar to approach in MW template parameters
    - cumbersome, limited to fixed amount of values
- Alternative 1: one subobject per element (value, pos.)
- Alternative 2:
  - store elements as property values without order
  - store whole list as Text in another property value

# Unsupported datatypes

## **Many types of data not supported in SMW.**

- Approach 1: Develop new datatype in PHP
  - Not difficult, yet rarely done
- Approach 2: Use existing datatype
  - Some datatypes can be customised (e.g., with units)
- Approach 3: “Implement” new datatype using input and output templates
  - Not often a good idea; template abuse

# Questions and Answers

